```
%  [Version of 27/10 2013]

/*  Difference lists

A difference list - representing a list [x1,...,xn] as a pair
                      [x1,...,xn|Rest]  Rest

Often convenient to join them in a term, e.g. [x1,...,xn|Rest]-Rest

Appending difference lists
  The concatenation of  X Y  and  Y Z  is   X Z   */

append( X, Y, Y, Z, X, Z ).

/* Eg.   ?- append( [a,b,c|A],A, [1,2|B],B, X, Y ).

% For the notation with -/2: */  append_dl( X-Y, Y-Z, X-Z ).  /*
.............................................................

Straightforward & inefficient list reversal (quadratic)

reverse_n([],[]).
reverse_n([X|Xs],Zs) :- reverse_n(Xs,Ys), append(Ys,[X],Zs).


Let us represent the second argument as a difference list */

reverse_d( Xs, Zs ) :- reverse_d( Xs, Zs, [] ).
reverse_d( [], Ys, Ys ).
/*
reverse_d( [X|Xs], Zs, Ys) :- reverse_d( Xs, Z1, Y1),
        append( Z1,Y1, [X|Y2],Y2, Zs,Ys ).

Now efficient (linear)

  From the definition of append/6 we have  Zs=Z1, Ys=Y2, Y1=[X|Y2].
  In other words, append/6 just unifies these three pairs.
  We may _unfold_ the call to append/6, obtaining
*/

reverse_d( [X|Xs], Zs, Ys ) :- reverse_d( Xs, Zs, [X|Ys] ).
/*
We obtained the previously discussed linear reverse
.............................................................

DANGER with difference lists:  here it may happen that correct
unification (with occur-check) is necessary.
E.g.  ?- empty( [a|T]-T ).    for empty/2 declared below
.............................................................
*/
```

```
% QUEUES as difference lists

empty( Q-Q ).

% deq( El, Q1, Q2 )  - El is the first element of queue Q1,
%                      Dequeuing it from Q1 results in Q2

deq( E, [E|Q]-R, Q-R ).

% enq( El, Q1, Q2 )  - Q2 is a queue Q1 with El added

enq( E, Q-[E|R], Q-R ).

/* Example queries

  ?- enq( 4, [1,2,3|T]-T, L-R ).
  T = [4|R],
  L = [1, 2, 3, 4|R].

  ?- enq( 4, [1,2,3|T]-T, Q ).
  T = [4|_G692],
  Q = [1, 2, 3, 4|_G692]-_G692.

  ?- enq( 4, [1,2,3|T]-T, Q ), deq( X, Q, Q1 ).
  T = [4|_G762],
  Q = [1, 2, 3, 4|_G762]-_G762,
  X = 1,
  Q1 = [2, 3, 4|_G762]-_G762.

Here we construct a "list with negative length"

  ?- deq( X, A-A, Q ).
  A = [X|_G686],
  Q = _G686-[X|_G686].

Adding an element to it results in empty list

  ?- deq( X, A-A, Q ), enq( a, Q, Q2 ).
  X = a,
  A = [a|_G744],
  Q = _G744-[a|_G744],
  Q2 = _G744-_G744.

  ?- deq( X, A-A, Q ), enq( a, Q, Q2 ), enq( b, Q2, Q3 ).
  X = a,
  A = [a, b|_G811],
  Q = [b|_G811]-[a, b|_G811],
  Q2 = [b|_G811]-[b|_G811],
  Q3 = [b|_G811]-_G811.

Try a list with length -2

 ?- deq( X, A-A, Q ), deq( Y, Q, Q2 ), enq( a, Q2, Q3 ), enq( b, Q3, Q4 ).

Taking an alement from queue Q1 before it is built

  ?- deq( X, Q1, Q2 ).
  Q1 = [X|_G683]-_G680,
  Q2 = _G683-_G680.
```

```
  ?- deq( X, Q1, Q2 ), enq( a, A-A, Q1 ).
  X = a,
  Q1 = [a|_G744]-_G744,
  Q2 = _G744-_G744,
  A = [a|_G744].

Two-directional information flow.
Dequeuer passes a constant b to the enqueuer:
Simpler version
  ?- enq( X, [1|T]-T, Q1 ), deq( 1, Q1, Q2 ), deq( b, Q2, Q3 ).

  ?- enq( f(a,X), [1|T]-T, Q1 ), deq( 1, Q1, Q2 ), deq( f(Y,b), Q2, Q3 ).
  X = b,
  T = [f(a, b)|_G406],
  Q1 = [1, f(a, b)|_G406]-_G406,
  Q2 = [f(a, b)|_G406]-_G406,
  Y = a,
  Q3 = _G406-_G406.

*/
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Some experimentation
% loop/0 runs in a constant size memory, but loop/2 does not
%                                        (with similar queries).
% Why?  The term from the query is kept.

loop :- empty( Queue ),
        enq( a, Queue, Queue_a ),
        enq( b, Queue_a, Queue_b ),
        enq( c, Queue_b, Queue_c ),
        loop( Queue_c, 0 ).

loop( Queue, N ) :-
        enq( N, Queue, Queue1 ),
        N1 is N+1,
        deq( _X, Queue1, Queue2 ),
        loop( Queue2, N1).
```